SR No	Name	Date	Sign
1	Tokenizing a file.		
2	Implementation of lexical analyser using Lex tool.		
3	Study the Lex and Yacc tool and evaluate an arithmetic expression with parentheses, unary and binary operators using Flex and Yacc (calculator).		
4	Using JFLAP, create a DFA from a given regular expression.		
5	Create LL(1) parse table for a given CFG and hence simulate LL(1) parsing.		
6	Using JFLAP, create SLR(1) parse table for a given grammar. Simulate parsing and output the parse tree in proper format.		
7	Write functions to find FIRST and FOLLOW of all the variables.		
8	Using JFLAP, create SLR(1) parse table for a given grammar. Simulate parsing and output the parse tree in proper format.		

Hardware and Software Requirement

1. Software Requirement:

Turbo C / C++

compiler. Download

from following links:

http://www.megaleecher.net/Download_Turbo_For_Windows

LEX tool--flex-2.5.4a-1.exe

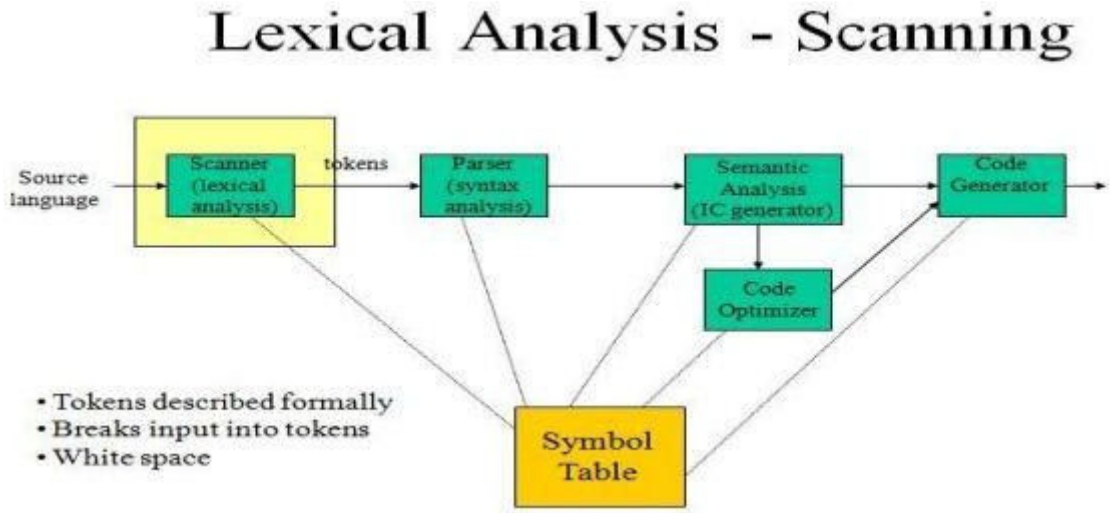
YACC tool--bison-2.4.1-setup.exe

JFLAP Tool for Automata - www.JFLAP.org

Practical 1: Tokenizing a file.

Aim: (Tokenizing). A program that reads a source code in C/C++ from an unformatted file andextract various types of tokens from it (e.g. keywords/variable names, operators, constant values).

Description: Lexical analysis is the process of converting a sequence of characters (such as in acomputer program of web page) into a sequence of tokens (strings with an identified
"meaning"). A program that perform lexical analysis may be called a lexer, tokenize or scanner.



Token

A token is a structure representing a lexeme that explicitly indicates its categorization for the Purpose of parsing. A category of token is what in linguistics might be called a part-of- speech. Examples of token categories may include "identifier" and "integer literal", although the set of Token differ in different programming languages. The process of forming tokens from an input stream of characters is called tokenization. Consider this expression in the C programming language:Sum=3 + 2;
Tokenized and represented by the following table:

Lexeme	Token category
Sum	"identifier"
=	"assignment operator"
3	"integer literal"
+	"addition operator"
2	"integer literal"
;	"end of the statement"

Source code:

```
import nltk
 # import RegexpTokenizer() method from nltk
 from nltk.tokenize import RegexpTokenizer
# Create a reference variable for Class RegexpTokenize
r tk = RegexpTokenizer('\s+', gaps = True)
# Create a string input str = "I love to study Compiler code in Python"
# Use tokenize method
 tokens = tk.tokenize(str)
print(tokens)
```